COLORCUE

VOLUME VI
NUMBER 1

A BI-MONTHLY PUBLICATION BY AND FOR INTECOLOR AND COMPUCOLOR USERS


```
EI              ;ENABLE INTERRUPTS
LXI     H,0     ;ZERO H & L
DAD     SP      ;ADD SP ADDRESS
SHLD    FCSSP   ;SP STORED HERE
LXI     SP,STACK;SET UP RUNNING STACK
CALL    SETUP   ;WHICH BASIC ?
LXI     H,LINBUF;CLEAR ALL BUFFERS
CALL    CLRBUF      GTPRM
LXI     H,PRTBUF    GETS THE PRINTER PARAMETERS
CALL    CLRBUF
CALL    CLRMEM      LEAR RAM FOR TEXT FILE
MVI     A,0C3H      SER JUMP
STA     81BFH               FPTR    EQU
LXI     H,EXIT  ;RE-ENTER WITHOUT CLEARING MEMORY
SHLD    81C0H       SCRIPT PRINTER PROGRAM
MVI     A,0C0H  ;DEFAULT BAUD RATE
STA     RATE    CALL    GTPRM   ;GET PARAM FOR PRINTER
XRA     A       XRA     A       ;ZERO CHAR COUNTER
STA     KBDFL   STA     CCOUNT
                                ;SET ERROR STATUS
LXI     H,MSG01 ;TITLE          ;SET NO ERROR STATUS
CALL    OSTR    ;PRINT IT
CALL    DRIVE       T DRIVE NUMBER & DIRECTORY
CALL    VECTOR      CALL    OPEN    ;OPEN THE FILE
JMP     FILES       JC      ERR02   ;IF CARRY, IT'S NO GOOD

IPC3:   MVI     M,0     NSERT TERMINATOR
        RET     LXI         TEXT    ;TEXT BUFFER ADDRESS
STC             SHLD    FPB1+FBUF;SAVE TEXT BUFFER ADDRESS
CMC     INFO:   CALL    TYPSET  ;CORRECT TEXT ADDRESS IF 'DOC'
RET     SETUP FILE AND PRINTER PARAMETERS
;       GETS INFORMATION FROM COMMAND LINE BUFFER
         'D RATE)
INFO:   LXI     H,BUFFER;POINT AT BUFFER

PUSH    PSW         ;JUMP VECTOR #31
PUSH    H           ;CONTAINS TEXT BUFFER ADDRESS
LXI     H,FPB1+FTYP
MVI     A,'D'   ;FIRST LETTER OF 'DOC'
CMP     M       LXI     D,FPB1  ;POINT AT INPUT FI
POP     H       LXI     B,DEFAULT;POINT AT DEFAUL
JNZ     TYP01   CALL    PFSPC   ;PARSE FILE SPEC
PUSH    D       JC      ERR02   F CAR  ERROR
LXI     D,0040H ;ALLOWANCE FOR S IBE  C ILE
CALL    SUBHD
SHLD    FPB1+FBUF;LOAD ADDRESS FOR 'DOC' FILE
POP     D
POP     PSW             INPCRT  EQU     81C5H
RET                     CALL    RWSEQI  ;REWIND INPUT FILE

KBDFL   EQU     81DFH
READY   EQU     81FFH
KBCHA   EQU     81FEH
LCOUNT  POP     D
        SHLD    FPB1+FXBC;SAVE BUFFER SIZ
        LXI     H,FPB1  ;RECALL BUFFER AD
GTPRM:  CALL    SCND    ;SCAN FOR DIGITS
        RC              ;NO DIGITS
        CALL    GVAL    ;GET VALUE IN (A)
        CALL    SETBR   ;CONVERT FROM TABL
        STA     RATE    ;RECALL IN PRINTER
IPC2:   MVI     A,20H   ;BLANK TO EI
        CALL    LO      ;TO THE SCRE
VECTOR: MVI     A,31    ;SET UP VE
        STA     KBDFL   ;TO INPCRT
        STA     MVI     A,0C3H  ;JUMP COM
        LXI     INPCRT
        STA     MOV     H,KBDINP;INPCRT JU
        LXI         INPCRT+1;TO KEYBOA
        SHLD    ;HOLDS NUMBER OF JU
SUM!Cl!y,       KEYBOARD STATUS
        ;KEYBOARD  ;CURSOR L
                    ;DO IT
FPB     FILE PARAMETER BLOCK AS YOU W
                    ;CONTINUE
        FILE TYPE
        ;BUFFER POINTER FOR
        ;BYTE COUNT FOR TRA
        ;BUF PNTR FOR SEQU
        INPUT FPB ADDRESS
        MVI     M,0     ;ZERO COUNTER
INPCOM: CALL    OSTR    ;ADDRESS IN CALLIN
        CALL    RESET   ;RESET IF ERROR
        LXI     H,BUFFER;POINT AT BUFFER
        CALL    KEYIN   ;READ FROM KEYBOARI
        CPI     13      ;IS IT CARRIAGE RE
        JZ      IPC3    ;YES, GO PROCESS CI
        CPI     26      ;IS IT BACK-SPACE
        JZ      IPC     ;YES, GO PROCESS BS
        MO      M,      ;STORE CHARACTER
        IN      H       ;INCREASE POINTER
        MOV     A,L     ;TEST LO BYTE OF P
        CPI     40H
        JZ      INPCOM  ;RESTART IF TOO BI
        LXI     H,KBDBUF;POINT TO COUN
```

Colorcue

VOLUME VI, NUMBER 1 JANUARY/FEBRUARY 1984

CONTENTS

COVER: A montage of SCRIPT by Myron Steffy.

EDITOR: JOSEPH NORRIS COMPUSERVE: 71106, 1302

COLORCUE is published bi-monthly. Subscription rates are US$18/year in the U.S., Canada, and Mexico (via First Class mail), and US$30 elsewhere (via Air Mail). All editorial and subscription correspondence should be addressed to COLORCUE, 19 West Second Street, Moorestown, NJ 08057, USA. (609-234-8117) Every article in COLORCUE is checked for accuracy to the best of our ability but is not guaranteed to be error free.

A CALENDAR PROGRAM is available from Christopher Zerr at 14741 NE 31st Unit, Bellevue, WA 98007. Written in Basic, it will print a calendar for any year neatly on an 8½''x 11'' paper through your printer, or on the CRT. Mr. Zerr will make a copy on your submitted, formatted CD disk, or send you a printout of the program. Please include $2.00 either way for packaging and mailing.

ROBOT: "Scorpion" is a 9" x 12" mini-wonder that resembles a NASA Lunar Landing Module built from an advanced Erector Set. It is powered by a 6502 CPU, 8K-bytes of EPROM and 2K-bytes of RAM. Two 6522 chips provide 32 I/O lines and four programmable timers. It has sensors to detect obstacles in its path, a two-axis optical scanner, with 1.5 degrees of scan per step, which will move over a 300 degree plane both vertically and horizontally.

Visual patterns can be displayed on a monitor. Additional hardware includes sensing bumpers, loudspeaker, two ground tracks, two "eyes" and four motors - two of which are drive wheels. Operating from a 12 volt DC power supply, "Scorpion" may be programmed through any RS232 bus. As a kit, with complete assembly and programming instructions, it is priced at $660.

Rhino Robots, Inc. PO BOX 4010, Champaign, IL 61820. 217-352-8485.

from the Editor's Desk

"...we do it all for YOU!"

Welcome to another volume of Colorcue. The magazine has been brought to maturity by the dedication of Ben Barlow and David Suits, our previous editors. The new staff can only hope to equal their success, and with a little help from our friends - you the subscribers - we have a good chance

We have some ideas for the coming year but it is only your ideas that can give Colorcue a meaningful vitality. While the number of subscribers experiences a gradual decline, the decrease in membership provides an opportunity for greater responsiveness to individual needs, and we hope you will assist in bringing this about by increasing your communication with us.

Many of you are in possession of splendid material for an article which, for one reason or another, never saw the light of day. The editorial staff can help! We will take your rough drafts and expand them, collaborating with you on the final copy. If that seems too forbidding then throw caution to the wind and ask a few questions. We can probably put you in contact with someone who has an answer. This simple exchange can lead to good articles; you may be sure someone else has wanted the answer too. It may help you to know that our readership contains those with all levels of proficiency, from beginners in Basic to accomplished assembly language hackers. Colorcue serves us all.

There are users involved with other computers, or thinking about "moving on." It would be valuable to hear how you are choosing your next machine. What will you be looking for, and how will your experiences with CCII affect your choice?

Some of us are privileged to be near active user groups and know and share with others on a regular basis. Many, many more are isolated and working on their own. Let this be the year for knowing one another better. The comment I have heard most often, since I became editor, has been an expression of desire for more interaction with other CCII users. Several authors have told me they wish they could hear more from readers - some response to an article, or just a note to share ideas or ask questions. The hobby belongs to us all, and you'll find most authors pleased at your responsiveness.

We continue to be impressed with the savings offered by THE COMPUTER SHOPPER. If you spend more than $50 each year on some kind of computer supplies you will spend nothing for this magazine. There are interesting things popping up in its pages from time to time that you won't read about anywhere else.

Several requests have been made in the past for your network mailing address. A list of subscriber network numbers would be a valuable asset for isolated CCII users. We are repeating the request, and for those of you who are curious but don't know how to begin we have included an introduction to modem communications.

Colorcue is offering some new services to readers which are described in this issue. They are there for you to use. We are prompt with our correspondence and we can guarantee every reasonable effort will be made to provide a useful response to your questions and comments. Take note of the SOURCEBOOK, to be printed as VOL VI, No. 3. and the pamphlets we have been preparing for specific applications, such as using the modem and FORTH.

We will try, too, to keep you informed about user group activities. You can become part of a user group by joining yourself, even if you cannot attend meetings. A $10 membership fee provides access to interesting newsletters and often a valuable free disk library. So make a few waves and see what a difference a little risking can make. It's going to be a good year!

Joseph Norris

LINKUP is a new magazine dedicated to network users and small computer communications in general. A monthly, printed in Minnesota, LINKUP will contain the latest information on data bases —new and old, reviews of pertinent hardware and software, protocol descriptions (!), book reviews, news of upcoming shows, user group notes, and tips on using telephone lines at minimum cost. Charter subscription is $19.83 for 12 issues. Newstand price is $2.95/copy. Credit card orders may be phoned to 1-800-543-1300, or write to LINKUP, PO Box 26345, 3938 Meadowbrook Road, Minneapolis, MN 55426.

FORTH COMPUTER: The Juniper Ace 4000 is a dedicated FORTH computer with keyboard (no monitor or media storage) for $175. It may be used as an intelligent programmable control for AC and DC devices with a suitable interface. Using the Z80 microprocessor at 3.25 MHz , the Juniper uses standard FORTH-79 in ROM, with 51K-bytes maximum RAM capacity. The computer may be connected to monitors and displays 32 columns by 24 lines. Low resolution graphics are available. Storage is by means of a cassette recorder. A FORTH programming manual is also available at $14.95. Computer Distribution Associates. 17 South Main Street, Pittsford, NY 14534.

COMPILING BASIC

—Part Two

Peter Hiner
11, Penny Croft
Harpenden
Herts, AL5 2PD
ENGLAND

In Part One we looked at how some of the simplest BASIC statements are handled by the Basic Compiler (FASBAS) and at the same time noted how these statements are handled by the Basic interpreter resident in ROM. In this part we continue to compare the Basic compiler and interpreter but consider more complicated functions.

We have looked at evaluating mathematical functions like A+B-C from left to right, but in contrast the function A+C*C requires that B should be multiplied by C before A is added. This is achieved by assigning priorities to the arithmetic operators, as described in the CCII Basic Manual. The order of priority is such that the contents of the brackets are evaluated first, followed by the power function, followed by multiply and divide, etc. Within a pair of brackets the same priority sequence is applied, and if two operators have the same priority, then evaluation is from left to right. Both the interpreter and the compiler use the same fundamental technique for handling evaluation, which is to compare the priority of the next operator with that of the one currently being handled, and if the next has higher priority, then all current information is pushed onto the stack to be retrieved after the higher priority function has been dealt with. So in our example of A+B*C, the procedure would be as follows. Move the contents of variable A to the Basic Accumulator, and when the + sign is encountered, move the contents of the Basic Accumulator to stack. Move the contents of variable B to the Basic Accumulator and then look at the next operator. So far the operation is the same as for the simple A+B evaluation.

If the next operator were an end of the line marker then everything would be set up ready to pop variable A from stack to registers BC and DE, and to proceed with addition (as described in Part One.) But in this case the next operator has a higher priority, so we must push the + instruction and the contents of the Basic Accumulator (i.e. variable B) to stack.

Now we move the contents of variable C to the Basic Accumulator and check the next instruction, which is the end of the line. So we can go ahead with multiplication, popping variable B from stack to registers BC and DE and calling the multiplication subroutine. This leaves the result B*C in the Basic Accumulator. Now we pop the previous instruction (+) from stack and compare its priority with that of the next instruction (end of line.) The comparison tells us to perform the addition next, so we pop variable A from stack to registers BC and DE, and call the addition subroutine, giving the result A+B*C in the Basic Accumulator. If the Basic statement had been longer, with another instruction (such as *D) having higher priority than addition, then we would have pushed the + instruction back onto the stack, pushed the contents of the Basic Accumulator (B*C) to stack and carried out *D next.

So how do we know when we have finished? The answer is that before we start an evaluation we push a zero value to stack, and we treat this just like the other operators (+, *, etc.). The zero value has lowest priority, equal to that of an end of statement or end of line marker, so it will not be dealt with until we have completely evaluated all the other mathematical functions. Then it will cause us to return from the evaluation subroutine to the routine we were in previously (e.g. return to a PRINT routine with the result of evaluation in the Basic Accumulator).

This is a convenient point to pause and consider the overall structure of Basic, which influences the design of both an interpreter and a compiler. We can identify three classes of Basic instructions, which I will call command, sub-command and operator.

Commands can be simply defined as those instructions which can appear as the first instruction in a line, and which include GOTO, GOSUB, PLOT, PRINT, IF, INPUT, etc. This class also includes LET, which never appears explicitly in our version of Basic, but is implied in statements such as A=B. These commands immediately put limits on the structure of the rest of the Basic statement. For example GOTO or GOSUB must be followed by a line number and then an end of statement or end of line marker. PLOT must be followed by a mathematical expression (variables, constants, etc.), and after that either an end of statement/line marker or else a comma to indicate that more PLOTS are required.

A command can therefore be used to direct the interpreter or compiler to a routine specifically designed to handle all the types of structure which are allowed in the rest of the Basic statement. The GOTO routine wil automatically treat the next character(s) as a line number. The PLOT routine will automatically treat the next character(s) as a mathematical expression to be evaluated and plotted, and will look for a comma to cause it to loop back and perform the same task again.

"We can identify three classes of Basic instructions...."

Sub-commands are instructions like THEN, TO, STEP, etc., which can only appear in statements following specific commands. Therefore they will be handled within the routines designed to handle the commands that must precede them. For example the IF routine includes a routine to handle THEN.

Operators are all those instructions which can appear within an expression that requires evaluation. This class includes the mathematical and logical operators like +, -, AND, OR, =, SIN, COS, etc., and also some less obvious operators such as PEEK and CALL.

The importance of this distinction between commands and operators is that this concept allows for a very simple structure within an interpreter or compiler. The initial command is used as a key to a table of addresses for the routines which handle commands. When the appropriate command routine has been entered, it will be used as the main program for interpreting the rest of the Basic statement. The command routine will call other subroutines as required, and in particular will call an evaluation subroutine (which I will name EVAL) to evaluate any combination of variables, constants, or operators that may be present between the command and the end of the statement (or other termination point such as a comma or a sub-command). The EVAL subroutine must handle this evaluation completely and leave the results set up in a suitable form for immediate use by the command routine. Evaluation includes not only mathematical functions, but also all forms of string and string functions, and all forms of comparison (for example in the statement IF A = 2 THEN..., the portion A = 2 is evaluated as a comparison and results in a value of -1 in the Basic Accumulator if true, or 0 if false).

The EVAL subroutine is extremely powerful and is central to both interpreter and compiler. There are major differences in implementation of the EVAL subroutine in the interpreter and compiler, but both apply the same fundamental principles. We have already looked at the principles of evaluation from left to right and in priority order. Now I will describe evaluation of brackets, and in particular nested brackets.

When a bracket is opened, evaluation of other parts of an expression must be suspended (everything pushed onto stack) until the contents of the bracket have been evaluated as this has the highest priority. To avoid any confusion in priority order between items inside and outside the bracket, the EVAL subroutine treats the contents of a bracket as a completely new evaluation, and to do this the subroutine calls itself. Subroutines using this technique are described as being recursive, and great care must be taken to prevent their vanishing inside themselves like the fabled Oozlum bird. Each time a bracket is opened, EVAL will call itself, and it will then unravel itself as the brackets are closed again. Although this technique makes it difficult to follow through the flow of a program, it does have the virtue of making the program very compact (for example the core of the EVAL subroutine in the Basic interpreter is less than 500 bytes long). The EVAL subroutine in the compiler sorts out nested brackets and presents them in the required order for computation in the final compiled program, so that, for example, ABS(INT(COS(A))) is presented for evaluation in the order A, COS, INT, ABS.

"...of practical value in speeding up programs."

I mentioned above that EVAL must also handle strings and string functions. For a proper understanding of this subject we should start with some fundamental properties of strings. The interpreter stores the references to strings in the same area as it stores variables, and distinguishes them by adding 80H to the second character of the name (e.g. variable A is stored as 4100H and string A$ is stored as 4180H). The other 4 bytes of a numeric variable location contain its floating point value, but for a string these bytes contain references to the address at which the string itself is stored and its length.

If, for example, line 100 contains the statement A$ = 'OK', then the reference address will point to the location within line 100 at which the first letter of the string is stored, and the length will be 2 characters. If line 110 contains B$ = A$, the EVAL subroutine will be used to evaluate A$, and will return with an address in the first two bytes of the Basic Accumulator (80DE-F). This address will point to the four bytes of the A$ storage location containing the references to the address of the string itself (i.e. OK) and its length. Now B$ can be made equal to A$ by simply moving these four bytes to the storage location for B$, so that both A$ and B$ point to the same string.

If we then have a statement PRINT B$, the EVAL subroutine will return with the address of the four byte storage location for B$, and this time the information stored there will be used to find the string (OK) and to PRINT two characters.

The compiler makes use of the same string subroutines as the interpreter in the final compiled version of a Basic program. It saves some time by loading the string name (e.g. 4180H for A$) directly to the registers and then it uses the ROM subroutines to search for the six byte string location, which is stored dynamically (i.e. at run time) in exactly the same manner and the same memory area as in a normal Basic program. This is still faster than in Basic, because the compiler allocates a separate storage area for variables at compilation time and therefore the search for a string does not get slowed down by having to sort through a large number of variables.

Generally the most that I can hope is that this series of articles will satisfy your intellectual curiosity, but the next section may also be of practical value in speeding up programs. This description of string contatenation applies equally to the Basic interpreter and to compiled programs.

When strings are contatenated (as in A$ = B$ + C$), the resultant string is stored in the string manipulation area at the top of memory. If B$ equals 'GOOD' and C$ = ' MORNING', a string 'GOOD MORNING' will be created and the A$ storage location will contain a reference to the storage address and length of this new string. If we now have A$ = A$ + ', SIR', another new string will be created in the string manipulation space and A$ will point to this string instead. But the string 'GOOD MORNING' will be left as a piece of garbage until the string manipulation space has been filled. Then the interpreter will sort through the strings throwing out garbage and making available as much free string space as it can. This can happen at unex-

pected times during a program run and cause a complete suspension of other activities for up to several seconds (particularly if you have increased the amount of string manipulation space from the minimum allocation of 50 bytes, by using a statement such as CLEAR 500). If you set the string manipulation space at the smallest size that will allow the program to run without OS ERROR messages, then you will spread the effect, causing a lot of short interruptions, but the overall result will be a longer run time. For example you might cause 100 delays of 0.1 seconds (total 10 seconds) instead of 1 delay of 5 seconds.

You can force garbage collection at any time you want by putting in a statement like W = FRE(X$). A suitable time would be when you have a message on screen which will require some time for the user to read. Beware of collecting garbage when you want a keyboard response from the user, as the interpreter will not be able to accept a key entry via a normal INPUT routine, and the user may respond before the interpreter is ready.

The best solution is to minimize the number of string concatenation operations, which are slow operations in themselves, apart from the delays caused by garbage collection. It is faster to PRINT A$;B$ than to PRINT A$ + B$. When you really need to concatenate strings, do it all at once rather than as a series of statements repeatedly tacking bits on the end of a string.

I once wrote a program which included a subroutine for converting numbers from decimal to hexadecimal, dividing the decimal number by 4096 to generate the first character of a string, then dividing the remainder by 256 to get the second character and concatenating them to make a two character string, then dividing by 16 and so on until I had generated a 4 character string. Now I know why it was so slow!

In Part Three of this series of articles I will describe how the compiler deals with arrays and the Basic commands. □

[FASBAS is available from the author for $25US. Ed]

FOREIGN LANGUAGE DEPARTMENT:

The indescribable language, FORTH, has been issued in three different versions for Compucolor computers. One issue is available for the 3651. It is probably the most-avoided language invented, yet the experience of FORTH has much to add to one's programming education.

FORTH is a language that combines high-level and low-level language functions. It is fast, self-'compiling', may be used in 'immediate mode' during the learning process—just like Basic—and provides a programming grammer of unusual proportions. FORTH has a vocabulary of 'key' words—just as Basic does—that may be invoked to perform specific predetermined functions. These words are arranged in sequence, by the programmer, to provide an ordered set of instructions.

FORTH uses a 'stack' just as assembly programming, but in a more immediate way. It also uses 'reverse Polish' notation, much as the Hewlett-Packard calculators. As an example, here is a FORTH instruction line to add the numbers '3' and '5' and print the sum to the screen:

```
3 5 + .        ( Type (3) (space) (5) (space)
                (+) (space) (.) )
```

Here is what happens as FORTH reads the above line:

```
3              ( '3' placed on top of stack )

5              ( '3' moved down the stack and 5
                placed on top )

+              ( The symbol '+', which means
                "add the top two numbers on the
                stack". These two numbers are
                popped off the stack as they are
                read, and their sum, '8' is placed
                on the top of the stack. '+' is a
                FORTH command word. )

.              ( The symbol '.' means "print the
                top of the stack to the CRT. After
                this operation the stack is empty. )
```

FORTH gives you the power to make your own commands. If you feel more comfortable using the word 'PRINT' to print, then an instruction early in your program can make this possible. It is very easy to define a new word in FORTH. On a separate line, one types a ':' (colon) to indicate a new word definition. Next the name of the new word to be defined is entered, followed by the list of instructions the new word is to signify. Finally, a semicolon is entered to terminate the definition. Here is an example (note each element of the definition is separated by spaces):

Going Forth...

```
: PRINT . ;    ( This means "hereafter when I type
                 PRINT, perform the '.' function." )
```

Or I might want to make a word that adds and prints both:

```
: PRINTSUM + . ;    ( This means "hereafter when I
                      type PRINTSUM, add the top two
                      numbers of the stack and print
                      their sum on the screen." )
```

This introduction of 'new words' to FORTH provides the programmer with a set of instructions (subroutines of a sort) tailored specifically to his needs. You can actually design your own programming language and never use a single word of 'real' FORTH, once your own words have been defined. For this experience alone, FORTH is worth some investigation.

You may combine your own new words to make still another valid command. Here is a word defined to perform a carriage return and a linefeed:

```
: CRLF 10 13 EMIT EMIT ;    ( My command word is CLRF. )
```

The command word 'EMIT' is like the Basic 'PRINT CHR$(x)' word. Numbers on the stack will be printed in their ASCII character equivalent rather than as numbers. Now I may define a FORTH word which I will call 'GO', which will simulate line 20 in the following Basic program:

```
10 A=1 : B=6
20 C=A+B : PRINT : PRINT C : PRINT
```

First I define my special FORTH word, equivalent to line 20 -

```
: GO CRLF + . CRLF ;
```

And this is how I would simulate the Basic program above, in FORTH:

```
1 6 GO    ( Put '1' and '6' on the stack,
            then do "GO." )
```

Another consequence of the FORTH experience is that 8080 assembly programming takes on a fresh and creative aspect. Many stack manipulations of FORTH are transferrable to assembly routines. Since most of us don't use the 8080 stack fully, FORTH provides us with valuable insights for increasing the vitality of stack-related instructions.

I was captivated by the method FORTH uses to store disk files. There need be no disk directory since FORTH uses the REAd and WRIte disk routines, storing files in 1024 byte blocks directly to the uninitialized disk. When you want to call a program to the screen you do so by specifying what start block it may be found on. If you keep no records on paper about the storage location of programs, it can be interesting trying to find them again. Nevertheless, there is a quality of magic about the process.

We are probably inclined to stay away from things that are 'good' for us but, for me, the primary value of FORTH has come from the discipline of total submission to a new, strange and demanding programming pattern. It is frustrating at first to lose the facility we have in other languages as we plod through the elementary learning procedures all over again. But second and third languages build on our previous experiences and facility does seem to come more rapidly with each new language.

FORTH may not become your favorite form of relaxation, but it is just plain fun to play with - and it's a good way to rekindle your excitement at the keyboard. Furthermore it is inexpensive. The Rochester Users Group has a version, free for the price of a $10 annual membership (Gene Bailey, 28 Dogwood Lane, Rochester, NY 14625). Bill Greene has just released a version for CCII and 3651 at no charge (but PLEASE send him a formatted disk and about $5 to cover costs: 3601 Noble Creek Drive, Atlanta, GA 30327). There is little documentation from either source but you will not have too much trouble until it's time for more advanced procedures.

There are three tutorial books that can be recommended. (I needed at least two!) The most accessible is by Thom Hogan: Discover FORTH. Osborne/McGraw Hill, 1982. The second is a little more formal—by Paul Chirlian: Beginning FORTH. Matrix PUblishers, 1983. Both of these volumes are available at leading bookstores or computer stores. The third volume is very recent: The Complete Forth by Alan Winfield, published by Sigma/Wiley, NY, 1983. For me, this third volume was the most appealing since it spoke to my level of programming savvy—not too little, not too much, but just right. I was disappointed that none of them discussed assembly or debugging, but I'm not that far yet! While none of these volumes is specific to the CCII, you will find that the FORTH basic vocabulary is fairly constant among different versions and that few special commands are required. All you need to get started is one version of FORTH on disk and one of the above tutorials.

[COLORCUE is preparing an 'interface' pamphlet to help you get started with Bill Greene's FORTH. You may write for it; the price is $2.00. If you are interested in reading more about FORTH in COLORCUE, drop us a line.]

A SERIAL TO PARALLEL INTERFACE is available for $90.00 which connects to the RS232 output of the CCII and produces a parallel "Centronics" output for a peripher device. It works well with pen plotters, printers, paper punches, and robots. Handshaking is provided as well as 8 Baud rates. It may be ordered from Engineering Specialties. Phone 805-487-1665 for information (CA). [Also see Ben Barlow's do-it-yourself article in Vol IV No. 3, DEC/JAN 1982.]

A PROGRAM TO LOAD SRC FILES INTO COMP-U-WRITER

Myron T. Steffy

The COMP-U-WRITER series of word-processors provide three modules for converting their own brand of text files into a type labeled 'DOC' that may be stored with the usual FCS disc SRC files generated by the COMPUCOLOR DOS. The DOC file retains the first 40H bytes of the file as necessary instructions to COMPUWRITER for reloading. In this form, the file could not be read by the Screen Editor but could be printed by a program named 'SCRIPT'. The first 40H bytes were simply discarded and the remaining text loaded and printed. There has been no easy way of performing the converse operation, that is, loading an original SRC type file into the COMP-U-WRITER framework.

The routine labeled 'LDFCS' is a machine language method of supplying the missing information and then loading the SRC file into the COMP-U-WRITER system. It will operate with versions 3.4, 3.5 and the current 3.6 generally known as the 'EXECUTIVE'.

The SRC files generated with the Screen Editor will have their lines terminated with a carriage return and a line feed. If the file has had its lines justified by a program similar to 'SCRIPT', there will be extra spaces inserted between words to stretch out the lines to a uniform right margin. When 'LDFCS' is used to load the file into COMP-U-WRITER, the line feeds are automatically removed as they are not required. There is an option in LDFCS that will allow you to alter the file so that the COMPU-WRITER's justification mechanism can operate.

If this option is selected, the carriage returns are replaced by 0A0H when they occur singly. This code is a terminator used by COMP-U-WRITER at the end of a line prior to justification. When double C/R's are encountered, they are retained to provide paragraph spacing as originally intended. All of these operations are more or less predicated on a 60 character line which is all that will fit on the CCII screen. If the file has longer lines, LDFCS will still work but the lines may double back upon themselves on the screen.

Since 'LDFCS' uses some functions within COMP-U-WRITER itself, the latter must be in place prior to running 'LDFCS'. This is also true of the other three modules. The method of operation is very much like the other modules. Be sure to do a [COMMAND/RESET] before loading the COMP-U-WRITER. After entering the usual preliminaries, the date and the drive number, exit the program with [CPU/RESET]. Then do an [ESC D] and 'Run' LDFCS in the default drive, not the drive number selected for COMP-U-WRITER text files. It will come up with a heading 'COMP-U-WRITER FILE LOADER' and the option question whether you want the C/R's and extra spaces removed. This defaults to 'Yes' as will usually be the case to allow COMPUCOLOR's internal justification mechanism to operate. If the file is an assembly language source file or other material where ordered columns are desired, answer 'No'.

The program will then prompt you for the file name and type. If it is an SRC file, you must so state as the default is 'DOC'. If it should be a 'DOC' type, the file will be loaded intact and the option rendered inactive. After typing in the file title, press [RETURN] and in a second or so, COMP-U-WRITER's usual heading will appear with the file you have selected. All of the usual functions are available and the text may be treated as any other file. Files of either type may be concatenated and the loading will take place at the point where the cursor is located when you exit COMP-U-WRITER. However, make it a rule to place the cursor at the end of the residual text and move the appended file with the [MOVE/BLOCK] function of COMPUWRITER after loading. When L/F's and other unwanted characters are removed from the file, the text will be shortened somewhat. Under certain conditions, you may find vestiges of the original text at the end of the file just loaded. Usually exiting COMP-U-WRITER with [CPU/RESET] and re-entering with [ESC] [USER] will straighten things out. Extraneous garbage may be readily removed with the [DELETE/BLOCK] facility if necessary.

With a 32K system there will be something over 17,000 (decimal) bytes available for text. If the file you are loading exceeds your RAM capacity, a message will appear on the screen to that effect. Pressing [RETURN] will take you back to COMP-U-WRITER and you will find a partial loading of the text file. A file too large for the system will have to be edited in sections. Again, if the tail end of the text is garbled, exit with [CPU/RESET] and re-enter with [ESC] [USER].

.The source code for LDFCS that follows is set up for assembly at 4000H where it may be retained and re-used for multiple file conversions. If you do not have a Devlin RAM board at that address, you may move it into the normal RAM area by changing the ORG to 8300H. Then at the very end, change the line 'DBSIZ EQU 5F00H-DBUF' to 'DBSIZ EQU 8E00H-DBUF'. LDFCS will have to be reloaded each time it is used since COMP-U-WRITER uses that area for I/O buffers and will overwrite it. □

```
;    LOAD COMP-U-WRITER TEXT FROM FCS SRC and DOC FILES
;=============================================================

;Removes L/F's and C/R's and excess spaces from SRC files.
;       by Myron T. Steffy, Sun City, Arizona  3/15/83

        ORG     8300H

START:  CALL    SETUP   ;WHICH BASIC ?
        LHLD    8F60H   ;HAS TEXT START ADDRESS
        SHLD    RAMST
        MVI     A,1     ;SET FLAG FOR SPACE REMOVAL
        STA     LFLG
        LXI     H,TITLE
        CALL    OSTR
        CALL    GETANS  ;WANT EXTRA SPACES REMOVED ?
        CPI     'N'     ;SAY NO FOR ASSEMBLER SRC FILES
        JNZ     QUERY
        XRA     A
        STA     LFLG

QUERY:  LXI     H,8EFFH
        SPHL            ;RESET STACK PTR
        CALL    912AH   ;INIT FOR RE-ENTRY
        CALL    RESET   ;RESET DISK IF ERROR
        LXI     H,MSG01 ;FILE SPECS
        CALL    OSTR
        LHLD    VHLAD
        XCHG
        LXI     H,NOCUR ;MOVE CURSOR OFF SCREEN
        CALL    OSTR
        LXI     H,BUFFER;POINT AT BUFFER
        MVI     B,20
        CALL    9F47H
        MVI     M,0     ;REQ BY OPEN
        LDA     91C6H   ;LOAD ENTRY
        CPI     1EH     ;UNMODIFIED  SCRIBE ?
        CNZ     9E51H   ;NO. CALL SAVE CURSOR
        LXI     H,FPB
        MVI     B,38
        CALL    CLEAR   ;CLEAR FPB
        LXI     H,BUFFER
        LXI     D,FPB   ;INPUT FPB
        LXI     B,DFDOC ;DEFAULT TYPE(DOC)
        CALL    PFSPC   ;PARSE FILE SPEC
        JC      E02     ;IF CARRY THEN ERROR

        LXI     H,FPB   ;INPUT FPB
        MVI     A,0     ;OLD FILE
        MOV     M,A
        CALL    OPEN    ;OPEN FILE
        JC      E02

        LXI     H,DBSIZ ;SEQ DISK BUFFER SIZE
        SHLD    FPB+FXBC

        LXI     H,DBUF  ;ADDRESS
        SHLD    FPB+FBUF
        LDA     FPB+FTYP;FILE TYPE ?
        CPI     'S'     ;SRC FILE ?
        JZ      SRCFIL  ;LOAD FILE ATTRIBUTES
        LXI     H,FPB   ;DOC FILE SPECS
        CALL    RWSEQI  ;REWIND FILE
        LXI     H,8F00H ;PARAMETER AREA
        MVI     B,40H
DSPEC:  PUSH    H
        PUSH    B
        LXI     H,FPB
        CALL    GTBYT
        JC      ERROR
        POP     B
        POP     H
        MOV     M,A
        INX     H
        DCR     B
        JNZ     DSPEC

TXT10:  LXI     H,FPB
        CALL    GTBYT
        JNC     TXT15
        JNZ     ERROR
        JMP     TXT20

TXT15:  LHLD    8F60H   ;CURRENT LOAD LOCATION
        MOV     B,A
        XCHG
        LHLD    8F62H   ;HIGH RAM LIMIT
        CALL    CMPDH
        XCHG
        JC      TXT17   ;NOT FULL YET
        CALL    90D2H   ;CLEAR SCREEN
        CALL    9DF3H   ;ERROR MSG - RAM FULL
        CALL    RESET
        JMP     TXT20   ;RETURN TO SCRIBE

TXT17:  MOV     M,B     ;STORE INPUT BYTE
        INX     H
        SHLD    8F60H   ;CORRECT COUNT
        JMP     TXT10

TXT20:  LDA     FPB+FTYP
        CPI     'S'     ;SRC FILE ?
        JZ      ST01

TXT30:  CALL    90D2H   ;CLEAR SCREEN
        CALL    0A650H  ;RE-JUSTIFY TEXT
        JMP     9270H   ;RE-ENTER

SRCFIL: LXI     H,PARA  ;PARAMETER LIST
        LXI     D,8F00H ;PARAMETER AREA
        MVI     B,40H
```

```
        CALL    MOVDH   ;PUT IT IN PLACE
        LXI     H,FPB   ;NOW GET TEXT
        CALL    RWSEQI
        JMP     TXT10

EXIT:   MVI     M,0A0H
        LHLD    SAVADR  ;END OF CORRECTED TEXT
        DCX     H
        SHLD    8F60H
        JMP     TXT30

E02:    PUSH    B
        CALL    90D2H   ;CLEAR
        POP     B
        CALL    EMESS   ;EMIT ERROR MESSAGE
        CALL    RESET
        CALL    WAIT
        JMP     TXT20

ERROR:  CALL    90D2H   ;CLEAR
        LXI     H,MSG02 ;PRINT ERROR MESSAGE
        CALL    0A1EEH
        CALL    WAIT
        JMP     TXT20

;       REMOVE C/R AND EXTRA SPACES FROM TEXT

ST01:   LHLD    RAMST   ;START OF TEXT RAM
        PUSH    H       ;SAVE IT
        XCHG
        POP     H       ;GET IT BACK
ST02:   PUSH    D
        XCHG
        SHLD    SAVADR  ;END OF CORRECTED TEXT
        XCHG
        POP     D
        MOV     A,M
        ORA     A
        JZ      EXIT
        CPI     0AH     ;L/Fs NOT WANTED
        JZ      SKIP
        ANI     7FH     ;CONVERT TO ASCII
        STAX    D       ;PUT CHARACTER AT DE ADDRESS
        MOV     B,A
        LDA     LFLG
        ORA     A
        JZ      NEXT    ;IF ZERO, SKIP THE REST
        MOV     A,B
        CPI     20H
        JZ      SPACE
        CPI     0DH
        JNZ     NEXT
        MVI     A,0A0H
        STAX    D
        INX     D
        INX     H
        MOV     A,M
        CPI     0AH
        CZ      NOPAR
        MOV     A,M
        CPI     0DH     ;SECOND C/R ?
        JNZ     ST02
        DCX     D
        STAX    D       ;TWO C/Rs FOR A PARAGRAPH
        INX     D
        STAX    D       ;SECOND C/R
        INX     D
        INX     H
        JMP     ST02

SKIP:   INX     H
        JMP     ST02

NEXT:   INX     H
        INX     D
        JMP     ST02

NOPAR:  INX     H
        RET

SPACE:  CPI     20H     ;SPACE ?
        JNZ     NEXT    ;NO
        INX     H
        MOV     A,M
        DCX     H
        CPI     20H     ;TAKE OUT ALL BUT SINGLE SPACES
        JNZ     NEXT
        INX     H
        MOV     A,M
        JMP     SPACE   ;DO IT AGAIN

WAIT:   LXI     H,9E28H ;'HIT RETURN'
        CALL    0A1EEH
        CALL    09E48H  ;WAIT FOR CR
        RET

CLEAR:  XRA     A
        MOV     M,A
        INX     H
        DCR     B
        JNZ     CLEAR+1
        RET

GETANS: MVI     A,50H
        STA     READY
GETCHA: CALL    0024H
        JNZ     GETCHA
        RET
```

```
SETUP:  LDA     0001H   ;VERSION 8/79?
        CPI     0BAH
        RZ              ;NO,  6.78
        LXI     H,NEWTAB
        LXI     D,OLDTAB
        LXI     B,LENTAB
MOVE:   LDAX    D
        MOV     M,A
        INX     H
        INX     D
        DCX     B
        MOV     A,B
        ORA     C
        JNZ     MOVE
        RET

;       SYSTEM ADDRESSES(6.78)

OLDTAB: JMP     262DH   ;EMESS
        JMP     3077H   ;PFSPC
        JMP     26A5H   ;RESET
        JMP     2DABH   ;OPEN
        JMP     30C6H   ;RWSEQI
        JMP     322CH   ;GTBYT
        JMP     33F4H   ;OSTR
        JMP     3453H   ;CMPDH
        JMP     343BH   ;MOVDH

        LENTAB  EQU     $-OLDTAB

;       SYSTEM ADDRESSES(8.79)

NEWTAB:
        EMESS:  JMP     0AD6H
        PFSPC:  JMP     14ADH
        RESET:  JMP     0B48H
        OPEN:   JMP     11E1H
        RWSEQI: JMP     14FCH
        GTBYT:  JMP     1662H
        OSTR:   JMP     182AH
        CMPDH:  JMP     1889H
        MOVDH:  JMP     1871H
```

```
NOCUR:  DB      3,64,0,239
DFDOC:  DB      'DOC'

TITLE:  DB      15,13,10,10,17,9,9,'COMP-U-WRITER FILE LOADER'
        DB      19,13,10,10,'REMOVE L/Fs  and EXTRA SPACES ? '
        DB      20,'(DEFAULTS TO YES) ',18,239

MSG01:  DB      13,10,10,19,'FILE NAME ',17,'(SRC or DOC) '
        DB      21,'DEFAULTS TO DOC ',18,239

MSG02:  DW      7290H
        DB      'INPUT ERROR ON DOCUMENT FILE',0

PARA:   DB      66,0,60,0,10,0,0,0,128,37,1,0,8,16,24
        DB      32,40,48,255,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DB      0,0,0,0,0,255,0,0,0,0,0,0,80,0

FPB     EQU     80F7H

READY   EQU     81FFH
VHLAD   EQU     81D2H
LFLG:   DS      1
RAMST:  DW      1
SAVADR: DW      1

EMFN    EQU     15H     ;MISS FILE NAME ERROR CODE

;FPB OFFSETS

FTYP    EQU     08
FBUF    EQU     32
FXBC    EQU     34

BUFFER: DS      20
        ORG     ($+127)/128*128
DBUF:   DS      0
DBSIZ   EQU     8E00H-DBUF

        END     START
```

We note with sadness the passing of Myron Steffy who died shortly after Christmas. Many of you will remember his steady stream of stimulating articles in Colorcue and Forum, particularly the evolution of SCRIPT, his word processor. Many others of us have known him as a supportive, knowledgable friend and helper. Myron was the first to respond to a call of assistance, whether it be help with a personal programming problem or the need for an article to enrich a not-quite-complete journal edition. Although in his seventies, Myron was in the front rank of those working to expand the utility of the CCII. As a programmer he was thorough and articulate. We will yet see more from him as some later projects are brought to completion by his associates. It has been a fortunate thing to have him in the Colorcue community.

David Suits
49 Karenlee Drive
Rochester, NY 14618

BASIC FROM THE GROUND UP. David E. Simon.Hayden Book Co.; 219 pp.; pbk

The reader who wants to know a little bit about internal operations of the computer, as well as learn a high level programming language, would benefit from this book. It starts on an elementary level, assuming the reader knows little about computers or even algebra, and clearly and concisely develops a knowledge about BASIC to a great extent. The book covers many aspects of the language and gives good sample programs to follow along with. The examples and the text will help the reader develop good structuring techniques for his/her algorithms and programs. Exercises are included at the end of each chapter to review key concepts, and there are appendices providing ASCII codes, a glossary, and a brief summary of BASIC statements.

Although each concept does not have an individual program which illustrates it, there are several good programs to illustrate many points at once. Most of these are immediately adaptable for use on the Intecolor or Compucolor computer. The book does base its applications on a time-sharing system, and some programs would require a little modification (omitting PRINT USING, random file commands, etc.). The programs cover such subjects as sorts, list look-ups, function graphing, and other mathematical problems.

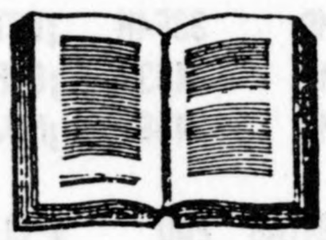
BASIC FROM THE GROUND UP is an excellent general work for someone who is seriously interested in learning BASIC for practical applications. The book concentrates on providing the text information on BASIC syntax and on descriptions of the BASIC vocabulary. It does not cover microcomputer-oriented BASIC as such, but it does provide enough knowledge of the language to enable the reader to program on microcomputers. This is not a book for someone who is simply interested in obtaining a list of programs to be copied and applied; instead, it gives the reader the tools necessary to be an effective BASIC programmer. It is an excellent reference work for BASIC.

DISCOVERING BASIC. Robert E. Smith,Hayden Book Co.203 pp,; pbk

This book begins as a brief introduction to BASIC, using many sample programs which illustrate points the author is currently covering. The work deals with the fundamental BASIC vocabulary without going into detail about the statements. It is intended to allow the reader to begin programming immediately, experimenting with each new concept as it is introduced. As the book progresses, the example algorithms become more complex but still presented at the neophyte level. Also, each chapter is summarized with a brief quiz which is graded by a corresponding BASIC program which the reader actually types in and runs.

Discovering BASIC contains over 50 programs, many of which could be entered directly into the Intecolor or Compucolor without modification. The last 70 pages are entirely devoted to algorithmic problems (solutions are given in the back of the book) that include computing pi, dice throwing, amortization scheduling and annuities. Although originally written with time-sharing in mind, the programs not directly applicable to Intecolor or Compucolor computers can be easily modified with a little scrutiny (ignoring MAT statements, entering BASIC from the time-sharing operating system, etc.).

In general, this book tends to stress algorithms as opposed to devoting much time to the finer points of BASIC. Within the text, the author has underlined key words to stress important points, increasing conceptual understanding. The immediate hands-on experience idea is also a very practical approach to the subject matter. However, the book is geared to time-sharing systems—not to microcomputers—and does not use many of the features of microcomputer-oriented BASIC (multiple statement lines, PEEK, POKE, etc.). Furthermore, the book does not go into computer fundamentals such as memory configuration, but instead takes a problem solving approach to learning. This would probably not be a good book for the person who wanted to learn BASIC in depth and to know about microcomputer BASIC in particular.



Introduction to 8080/8085Assembly Language Programming by J. N. Fernandez and R. Ashley. John Wiley and Sons, 1981. 303 pp. pbk

Here is a most excellent introductory book for those persons who wish to begin learning about 8080 assembly language programming. The book assumes only a minimum of experience with computers. (A little BASIC is fine.) The topics include binary, decimal and hexadecimal numbers, binary arithmetic, the ASCII code, the 8080 instruction set, pseudo operations, plus tutorials on writing 8080 programs for simple number crunching.

The book is a 'self-teaching guide'; it has numerous exercises which are to be done in the spaces provided. (Solutions are given, too) This is a good approach, since, after all, learning to write assembly language programs can be done only by actually writing them. And the large format of the book (6-3/4″ x 10″) provides plenty of space for working out the frequent exercises and samples.

The front cover photo is of an Intecolor 3600 computer. Alas, there is nothing in the book which addresses itself specifically to ISC machines. In fact, most of the exercises will have to be altered (in very simple ways) if they are to be actually entered and run on an ISC computer, since the RAM addresses used in the text are not RAM addresses on an Intecolor or Compucolor.

There is a lot the book does not cover, but an introductory text such as this is not meant to anyway. And all those features specific to ISC machines (such as ROM routines and color graphics) are, of course, not covered. The book is meant to introduce the reader to 8080 assembly language programming, independently of the particular machine one uses. And this it does very well. □

REFRESHER COURSE

You may create and read non-random files from BASIC. The only restriction is that the format of the file must be expressible in terms of the FILE 'N' and FILE 'R' statements. You may supply your own extension code.

```
Example:  FILE "N", "NOTES.PDQ",2,128,1

          FILE "N", "QUACK.SRC",1,256,1
```

To recover data from a .SRC file, created anywhere, you might do this;

```
100 FILE "R",1,"TEXT.SRC",1;2,64,1

120 GET 1;A$[64],B$[64]
```

This feature is very handy for accessing a data base in .SRC form, updated by some other program, and passing parameters to BASIC. You cannot extract numerical data directly since BASIC uses a special form for numbers, but you can retrieve a number in string form and use A = VAL(A$) to convert.

A typical example of use might be a catalog of items with a description, weight and price. This data is updated on a screen editor in strict format, but called by a BASIC program which writes invoices. (Expanded from a note in CUWEST, via FORUM.) □

GETTING STARTED WITH THE MODEM

Joseph Norris

It's a whole new world with a modem connected to your Compucolor II. What a wonderful way to 'reach out and touch someone' or dispel the 'Basic blues.' While the initial cost is significant, maintaining a modem connection with others is relatively inexpensive and a continuing source of entertainment.

With a modem you may communicate with other computers (not just CCII) through the telephone lines, or you may subscribe to a 'network' and utilize their facilities for work and play. A popular beginning network is CompuServe, which offers moderate resources at a low maintenance fee. This summary will describe the initial steps to establishing a modem and network connection for yourself, and define the approximate costs for putting it in operation.

You will need a modem. This device accepts output from the 'printer port' of the CCII and sends it over an ordinary telephone line. It also accepts signals from the telephone line and puts them on the CCII input port. (Both these ports are on the same connector.) Some lower-priced modems transmit only at 300 Baud while others add a 1200 Baud capability at a significantly higher cost. We use the Hayes Smartmodem(T), a 300 Baud device, which may be purchased for about $240. (300 Baud is fine for most purposes.) The modem 'input' connects to the MODEM or RS232 bus connector on the CCII. Unlike many other peripherals for the CCII, your modem can be used with most any computer you may purchase in the future.

The 'output' of the modem connects to a standard telephone outlet, usually in 'parallel' with a telephone handset which is used to dial numbers in the usual way. You will need to use the modem near such an outlet, or run one to your modem location. We bought an inexpensive ($9.95) telephone to use with ours for verbal communications (the modem does the dialing!).

Modems are available with a large variety of features, too many to describe here in detail. Some modems will store telephone numbers for you, dial automatically, turn on your computer if a call is received and you're not home, etc. Our recommendation is that you 'keep things simple.' CCII users have had good experiences with the Hayes and the CAT Novation modems. You will need to do some homework if you propose to add a 'fancy' device to the CCII.

You will also need some kind of 'terminal' software to enable your CCII to act like a terminal only. Comtronics has a nice program for this (TERMII) which adds some capabilities such as saving incoming data to disk for later use, and transmitting disk contents over the modem. TERMII costs about $70, and is available from CCII dealers.

At this point, with the connections made, you may communicate with another modem operator directly. The cost will be the telephone charges for the connection, just as you would pay for an ordinary telephone call. Assuming you had made arrangements with

SUBMITTING ARTICLES

You may submit material to Colorcue in a number of ways. Although we use a 3651, we can usually back up CD disks written with v8.79. If you use COMPUWRITER make a DOC file from your text and send that. You may submit SRC files as well. Modem owners may reach us through COMPUSERVE or by direct connection if a prearrangement is made by mail or telephone. For 3651 users, we can read MD, FD and DF disks (but not DM). If none of these methods is suitable, just send your text on paper. If program listings are a part of your material, please use a fresh ribbon to print them (white typewriter paper preferred) so we won't have to copy them and face the possibility of errors.

another party to talk at a specific hour, one of you will dial the other's telephone. You may speak to one another on the phone and by computer. With your terminal program running, you speak to him/her by typing on the keyboard. What they reply is printed on your screen. Both your message and the return message can be displayed on the screen—in different colors— so the entire dialogue is before you. You may also transfer files of various types. This means much of your transmission may be prepared in advance, thus lowering the expense of the actual connect time.

NETWORK CONNECTION. Possessing the equipment discussed so far, and using CompuServe as an example, here are the steps necessary to begin network communications. If you write a request letter to CompuServe they will send you a 'beginners kit' at a small charge. This kit contains general information about the network, a CompuServe log-on number for you to use and a 'password' that is exclusively yours. One hour of connect time is included in the price. Another easy way to begin is to buy, from RADIO SHACK, a TRS-80 Videotex Universal Sign-up Kit (#26-2224, $19.95). This kit contains a CompuServe number, a password, explanatory material, and includes one free hour of CompuServe time. (This is often the most inexpensive way to get a first membership.) Some modems come from the factory with 'access' packages enclosed, so, if you have the hand set and line connections, you can begin right away. Detailed descriptions of procedures are included. From either source, instructions are provided for continuing your membership beyond the first hour, if you wish. At 'log-on' time your modem will indicate to the network which Baud rate you are using. The hourly charge for 1200 Baud is higher than for 300 Baud.

CompuServe provides a number of interesting services, including a network mailbox, 128K of memory storage, user group bulletins, programs and games, and news items. There are 'hundreds' of data bases in this service, some requiring additional fees. For example, you may expand your access to Dow Jones reports, national news wire services and other more specialized information, including news releases by Federal agencies (which can be fascinating). Accumulated charges will be billed to you monthly by the network. If they are billed to Master Card or Visa charge accounts there is no billing charge (*).

CompuServe has installed many 'local' telephone access numbers across the North American Continent so that 'long distance' calls are not usually required, or at least available at reduced rates. Our CompuServe connection is only three miles away. A list of access telephone numbers will come with your information kit. The charges include a yearly membership charge, a 'log-on' time charge of about $6.00 per hour (if you use the service between 6PM to 5AM), billed in one minute segments, and the use of the telephone line from your local CompuServe exchange to the CompuServe computer complex (about $5/hr.) Since actual connect times can be very small, these charges are not a great burden. For example, we access our mailbox three times each week for a total log-on time of about 2 minutes. This includes the time I use sending messages as well as receiving them. Since messages can be prepared in advance, saved on disk, and transmitted from disk at maximum speed, little actual connect time is required.

"...a monthly expense of $12 will provide a goodly amount of entertainment."

Terminal programs will allow you to save all incoming data for later examination at your leisure. If you are playing a CompuServe game 'on-line' in real time, the 300 Baud rate, billed at one-fourth the cost of 1200 Baud, is the better method. All that time you spend 'thinking' on an open line costs money. Since we do not think faster at 1200 Baud, the 300 Baud rate and costs are the way to go. The only time 1200 Baud is cost effective is when long data files of many pages are being sent. COLORCUE goes to the typesetter by modem and at 300 Baud it's a bit costly. But this only happens once every two months and represents about 1/50 of total modem use for COLORCUE.

....*MORE BLUE SKIES*

Some primary objections to the CCII might be a) a limited software base (we don't have the rich variety that CPM or IBM/PC users have); b) only 63 characters per line (we can't have 120 column screen displays); c) 'poor' graphic resolution (no spectacular moving pictures); d) an inadequate disk storage system (not enough capacity, non-reliable operation); e) frequent failures from aging equipment (no service facilities, hard-to-get replacement parts.)

Colorcue has prepared a small users manual for connecting and operating the Hayes SmartModem using TERMII by Com-tronics. The price is $2.00. This manual details step-by-step procedures for those who don't want to unravel the puzzle for themselves. To summarize expenses, your initial investment will be about $350 for modem, handset, cables, software and CompuServe 'starter kit.' After that, a monthly expense of about $10 will provide a goodly amount of entertainment. The COLORCUE editorial office welcomes CompuServe mailbox communications, so you already have one place to 'call.' Here are addresses for two popular networks.

COMPUSERVE: 5000 Arlington Centre Boulevard, PO Box 20212, Columbus, Ohio 43220.

THE SOURCE: 1616 Anderson Road, McLean, Virginia 22102

* Note: For a detailed survey of networks available and the services they offer, see PERSONAL COMPUTING magazine, January 1984]

Sounds depressing, doesn't it! Yet, the fact is that I use my Intecolor equipment most of every day to work and play and I am not feeling deprived in any sense of the word. Sometimes I look with longing at ALT keys, 600 X 400 screen resolution and feel, frankly, a desire to escape to something new. It must be related to all the other feelings of disenchantment we sometimes feel—with our jobs, our spouses, our children, and.... ourselves. Yet, after spending any amount of time on another system, I'm always glad to get back. A closer look will show the picture isn't so bleek after all.

Tom Devlin has given us access to CPM and, with that, a good disk drive system. He has made satisfactory resolutions with the 63 character limitation of the CCII as a monitor for many, many programs. (CP/M was designed for a 64 character display, by the way.) With CPM programs and the supurb CCII utilities available, we have a software base more rich, and less expensive than most. (Have you ever added the cost of your software equivalent on CPM?)

Equipment failures are another matter. Compucolor (Intecolor) still provides factory service on the CCII. I have used it recently, accompanied with a carefully-worded plea for mercy, with satisfactory results. There are several fine service facilities still available from selected dealers and service stations across the country. (We'll provide an update on this for you very soon.) A major cause of CCII failure has been connected with the analog board. Tom Devlin has a 'saver' board for installation in the CCII which prevents one major failure mode. Write to him for information. We plan to print a review of this device in the next issue.

The graphics resolution we can't do much about, but if you have been keeping up with the spectacular releases in CCII game software you're not feeling too deprived.

Now to the blue skies..... with the current access to 4000H, we have a place to put, in ROM, disk handling routines for MD and FD standard drives in the 40 track, 128 byte/sector format. What we need is a System Chip that is modified to bypass CD disk handlers and jump to routines in the 4000H space (in ROM or RAM) for disk handling of MD (5″ standard 92K floppy drives. 4000H might also contain a CD to MD backup program for transferring existing files. The keyboard edge connector is a satisfactory interface for such a system. If bank selector boards are in use, Bank 0, under software control, can be assigned the disk handling function, with automatic power-up to these routines. True, some software modifications might be required for existing programs, but not much.

Our hardware and software experts might consider this as a new product venture. To operate a CCII with MD reliability and speed would shed a new light on our 'old, outdated' machines. One or two chips and some drives, that seem to be reaching very low price levels, would do it. (8″ drives are selling for as low as $150 each!) Let's have some response. Can it be done?!

QTZ

The following program, submitted by Tom Napier, is our first CUTIES in assembly language and must be entered with precision. It's great in a darkened room! Use [COMMAND/RESET] to stop, and reset Basic's pointers with [ESC] [W] before calling another program.

```
12 PLOT 6,3,12,3,20,10 : PRINT "DYNAMIC ELLIPSE DOODLER"
15 PLOT 3,25,15 : PRINT "BY TOM NAPIER" : GOSUB 300 : B=33282
40 POKE B,195 : POKE B+1,0 : POKE B+2,144 : PLOT 12
65 A=INT(65535*RND(1))
70 Z=CALL(A) : A=A+1 : IF A>65535 THEN A=A-65536
80 GOTO 70

100 DATA 229,33,0,0,66,14,127,83,30,127,205,128,144
110 DATA 205,0,145,124,181,194,10,144,225,201,256
115 REM
120 DATA 229,122,230,124,111,38,0,41,41,41,41,41,120
130 DATA 230,126,213,95,22,112,25,209,213,120,230,1
140 DATA 135,135,0,95,122,230,3,131,60,95,151,55,23,29
150 DATA 194,165,144,209,174,119,227,35,124,227,35,230
160 DATA 28,15,15,246,128,119,225,201,256
165 REM
170 DATA 229,120,167,31,47,103,121,31,0,111,35,25,84,93
180 DATA 122,167,31,103,123,31,111,9,68,77,225,201,256
190 REM
300 X=36864 : GOSUB 350 : X=36992 : GOSUB 350 : X=37120
350 READ Q : IF Q=256 THEN RETURN
360 POKE X,Q : X=X+1 : GOTO 350
```

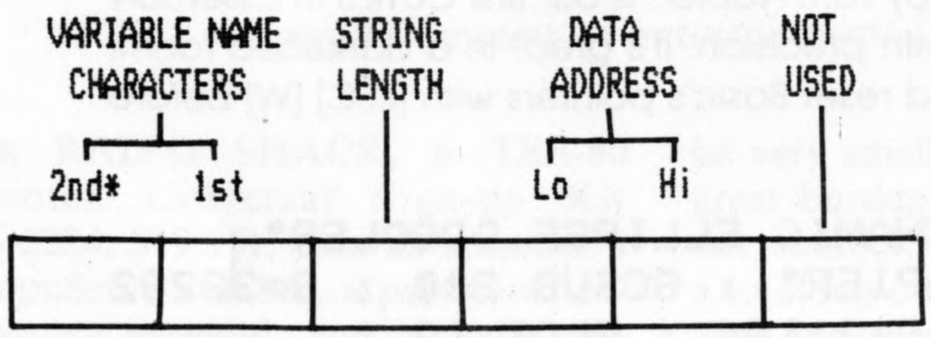
HOW BASIC STORES VARIABLES

Gary Dinsmore
32695 Daisy Lane
Warren, OH 97053

The Compucolor Programming Manual lists the address of the pointer to the start of Basic variables. This is a two-byte address at 32983 and 32983 (80D6H and 80D7H). If you use a debug program to extract this pointer address, you can look at the values in the variable storage area after a Basic program has been run, and while the variable data is still intact. You will find the variable names and other important information about them, stored in 6-byte groups.

Like most things having to do with the 8080 microprocessor, these bytes are stored "basackwards." To find the start of variable storage add 256 * PEEK(32983) to the value of the byte in 32982.

When you begin to examine the actual variable data you will need to know how to interpret it. The first two bytes are the variable name—second byte first. The first byte is coded to indicate whether the variable is a "string" or a numeric variable. A value greater than 128 indicates a "string variable", and the format in FIG 1. will be used.



*For 2nd character of variable name (ASCII), 128 = no 2nd character. Subtract 128 to get ASCII value of character

FIG 1.

If the value of the first byte is less than 128 then the variable is numeric. In this case, the value of the number stored in bytes 3, 4, 5, and 6 is a binary floating-point representation. Bytes 1 and 2 still represent the second and first characters of the variable name respectively. If a single letter variable name is used, then the first byte of the six bytes is a "0." The last four bytes (32 bits) of numeric value include a sign bit, 23 bits of floating-point precision, and 8 bits of mantissa.

Lets take a decimal analogy. The decimal number 132 can be represented in exponential notation as +1.32 E2. A binary number can also be represented in exponential form. In this case it will be +1.00001B E111B. We can interpret these two versions of the same number position by position.

First, decode the fractional parts of both numbers:

DECIMAL: +1.32 E2	BINARY: +1.00001B E111B
1 * 10^0 = 1	1 * 2^0 = 1
3 * 10^-1 = 3/10	0 * 2^-1 = 0/2
2 * 10^-2 = 2/100	0 * 2^-2 = 0/4
	0 * 2^-3 = 0/8
	0 * 2^-4 = 0/16
	1 * 2^-5 = 1/32

Next, decode the exponent for each number:

E2 in base 10 = 100 **E111B = 2^7 = 128**

Now multiply each number by the exponent:

1 * 100 = 100	1 * 128 = 128
3/10 * 100 = 30	1/32 * 128 = 4
2/100 * 100 = 2	
SUM = 132	SUM = 132

To see how this data fits into four bytes, lets draw a picture. This is how the six bytes for variable "X7" would look with the value 132 stored:

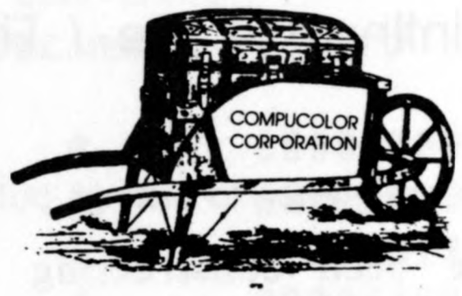
Byte	1	2	3	4	5	6
Binary	00110111	01011000	00000000	00000000	00000100	10001000
Hex	37	58	00	00	04	44
ASCII	'7'	'X'				

Lets rearrange the bytes in a more logical order:

Byte 2	Byte 1	Byte 5	Byte 4	Byte 3	Byte 6
01011000	00110111	00000100	00000000	00000000	10001000
'X'	'7'				
		Sign bit + 23 bits of fractional precision for value of variable. 1.fffffffffffffffffffffff (The '1' is implied.)			Add 127 to this byte to get the power of 2 for the fractional part, = a number from 127 to -128.

The following Basic program segment can be typed into the last few lines of any program and used to list the variables and dump their values. Remember, only variables

that have been assigned are listed. The variable "A" is assigned first and then used to transfer the values of the other variables into a form that can be interpreted and reported by Basic—without resorting to an elaborate decoding scheme. (I let Basic do it!) [If "A" is used as a variable in your program, substitute an unused variable for "A". Ed.]




```
0 CLEAR : A=0
1 REM This assigns A first so we can find it.

59999 END : REM Protective end statement.

64000 Q3=PEEK(32982) + 256 * PEEK(32983)
64001 REM List out all variables except arrays

64010 FOR Q1=Q3 TO PEEK(32984) + 256 * PEEK(32985) - 1 STEP 6
64020 IF PEEK(Q1) > 127 GOTO 64200 : REM Handle strings.
64030 PRINT CHR$(PEEK(Q1+1)); CHR$(PEEK(Q1));"  ";
64040 POKE Q3+2, PEEK(Q1+2) : POKE Q3+3, PEEK(Q1+3)
64041 POKE Q3+4, PEEK(Q1+4) : POKE Q3+5, PEEK(Q1+5)
64042 REM Transfer value to A.

64050 PRINT A
64060 NEXT : RETURN

64200 PRINT CHR$(PEEK(Q1+1));
64210 A= PEEK(Q1) : IF A>127 THEN A=A-128
64220 PRINT CHR$(A);"$  ";
64230 FOR Q2=0 TO PEEK(Q1+2)-1
64231 PRINT CHR$(PEEK(PEEK(Q1+4)+ 256*PEEK(Q1+5)+Q2));
64231 NEXT Q2 : PRINT : NEXT Q1 : RETURN
```

The SOURCEBOOK is coming.....

Colorcue's **SOURCEBOOK** will be printed in Vol VI, No. 3, May/June 1984. It is designed to be a compendium of all materials available for the CCII and 3651 computers. If you have any material available for these machines, here is your opportunity to let Colorcue readers know who and where you are. It is particularly important that dealers who have not advertised for a few months or more make their continued presence known. Please contact our office promptly if you wish to contribute materials. There is no charge in the **SOURCEBOOK** for advertisements or copy of any kind. Editorial discretion will be used and space allocated as it is available.

The planned contents includes in part: an index of Colorcue. Forum, Data Chip and as many other publications that are made available to us; a complete documentation of ASCII; a cross reference of printer commands among Epson, IDS, Okidata, and C. Itoh (Apple and NEC); comparative ROM listings for v6.78, v8.79 and v9.80; a descriptive catalog of currently available "commercial" software; user group disk holdings; RS232 definitions and "standards"; list of current hardware available for CCII and 3651; up to date dealer list and repair center list; bibliography of tutorial books applicable to the CCII and 3651; a directory of current Colorcue subscribers with network access numbers (you may have this information withheld upon written request.)

ASSEMBLY LANGUAGE PROGRAMMING

PART XIII Printing the File / Finishing Up!

Joseph Norris
19 West Second Street
Moorestown, NJ 08057

[This article makes reference to Listings published in Colorcue, Jun/Jul 1983.]

If you have been constructing SOURCE.SRC with these articles, you know by now that the text format is not particularly elegant. Words at the end of the first text line 'wrap-around' to the second - sometimes with inglorious division of syllables. Well, it's up to you to deal with that problem. My point is that the print routine won't be any better. What you see on the CRT is what you'll get. The important topic for our consideration is how to get those bytes to the RS232 port.

THE PRINT ROUTINE (See Listing V.) It is only appropriate to print text from an open file, so we begin by verifying an 'open' file status. Sending our text characters to the outside world requires a setting of the BAUD rate (rate of character bit transmission per second)[1], setting the number of stop bits, and feeding the characters to be sent to the operating system subroutine.

BAUD RATE GENERATOR. The Baud Rate is set by addressing Output Port #5 in the CCII and 3600 series computers. It expects to see a number between 01H and 40H (see BAUDTB, Listing V) corresponding to baud rates between 100 and 9600. In the fifth through tenth lines of module PRINT (Listing V) we get the ASCII number entered at the keyboard representing the selected Baud Rate, and subtract 30H to convert it to the correct decimal number, 1-7. What happens next may be new to some readers; we will change our decimal number (1-7) to the proper code number for setting Baud Rate using a table of equivalents we have installed in memory at address BAUDTB ('baud table').

We will use the system routine **ADHLA** to add the contents of the accumulator (our selected baud number code) to the HL register pair. This has the effect of indexing the HL memory pointer to the correct corresponding conversion number in BAUDTB. [In Fig. 1, examination of the binary equivalents makes it clear that we are setting one of seven bit switches, one for each Baud Rate available.] We move the converted number into the accumulator and send it to Port #5, which is internally wired to the baud rate generator in the computer. [2]

STOP BITS. In BAUDTB, bit 8 is shown as '0.' This bit sets the number of stop bits; '0' = one stop bit, '1' = two stop bits. We can 'set' this bit by adding 80H to our number code in A (the same as adding 10000000B, thus setting bit 8.) Before addressing BAUDTB, our program is already set at two stop bits because we executed a PLOT 15 in string CLR. Since the data in BAUDTB will have been transmitted to the printer just before printing, it will supercede any previous settings.

THE PRINTER SETUP STRING. A 'setup string', here, means a string of characters you may want to send to your printer to set characters/inch, margins, paper positioning, etc. The actual characters used vary from printer to printer but are usually Escape sequences. Here is a string I might use with my C. Itoh 'Prowriter' to set 12 cpi and 6 lines/inch index spacing -

```
SETPR:  DB  27,69,27,65,0 ; ESC E, ESC A
```

The '0' has been added as an 'end-of-line' code so I may send these bytes using **S1OUT**, with 'CPI 0' to detect the end of the string. However, if I replace the '0' in the string with '239', and set **LOFL** to '0EH' (output to serial port), I can use **OSTR** to send the bytes (see COLORCUE, FEB/MAR p13 and be sure to change **LOFL** back to 00H afterward.) It is helpful to keep GTCHA in the routine, even if a setup string is used, to allow for aligning paper, etc. before printing commences.

S1OUT, the Send Routine. The label means 'send one out' and the second character is the numeral one, not the letter 'I'. This routine requires that the E register contain the character to be sent, before CALLing. **S1OUT** will obey the 'clear to send' line (the 'handshake' line) on the RS232 port, waiting patiently (forever, if necessary) for the printer to OK the send (digital '0'). If there is no printer connected, the byte will be 'sent' into thin air and the routine will RETurn.

Different printers may require some modification of the send routine with respect to carriage returns (cr) and line feeds (lf). My printer may be programmed to do a cr-lf both when either character is received (in non-graphics mode.) Technically, a carriage return only moves the carriage to the left side of the paper. Only a line feed indexes the paper up one line. In lines 19 and 20 of module PRINT, I send a line feed (and cr, because of my printer characteristic) to print a space between printouts of the text.

The C register, in the print routine, counts the number of characters printed per line and the B register counts the total characters to be printed. We point to INBUF with HL, move each character into the E register, send it, and then adjust the counters and pointer until we are finished. At the end of the first line (when C=0) we branch to subroutine XX8 which indexes the paper for a new line. Here, the cr and lf are sent separately - as an example.

We said in Article XI, footnote [1], that printers interpret the ASCII characters between 1 and 31 differently than the Intecolor or Compucolor CRT. Fig. 2 is a table of the printer's interpretation with a brief explanation. While this chart shows the expected function of each code you must check your printer instruction manual for possible exceptions.

We have only three more modules to enter in SOURCE - those that provide the exit from our program to FSC, BASIC, and CRT mode. The BASIC exit is a routine I have used in these pages before, by M.A.E. Linden. With a 'D' in the H register, the operating system routine **ESC1** takes us back to

FIG 1. BAUDTB CONVERSION

===== POINTER =====		== ADDRESS ==		======= CONTENTS =======		
				Hex	**Decimal**	**Binary**
HL + (A=00H)	->	8507	BAUDTB:	00H	0	00000000
HL + (A=01H)	->	8508		01H	1	00000001
HL + (A=02H)	->	8509		02H	2	00000010
HL + (A=03H)	->	850A		04H	4	00000100
HL + (A=04H)	->	850B		08H	8	00001000
HL + (A=05H)	->	850C		010H	16	00010000
HL + (A=06H)	->	850D		020H	32	00100000
HL + (A=07H)	->	850E		040H	64	01000000

FCS (notice we are simulating the key presses 'ESC' and 'D')[3] if we have saved and restored the FCS stack pointer (which we have!) The CRT jump is not so elegant and may give unpredictable results. It should result in a blank screen - without the 'CRT mode' label, but, depending on what has taken place in the computer beforehand, may display a few extraneous characters. You **WILL** be in CRT mode, however. It would probably be best to restore the FCS stack before making this jump. You can add those program lines yourself, using FCSJMP as an example.

INITIAL ROUTINES. We begin by saving the FCS stack pointer for later use in exiting SOURCE.PRG. We also assign the beginning of our own stack [4]. This location is not to be left to chance, and if you have been lucky 'till now, your days are probably numbered. From its starting location, the stack works backwards toward 0000H, so adequate room must be provided for it. Normally numbers that go on the stack come off in equal quantity. A major exception occurs when a subroutine is terminated by a JMP instruction and not a RET. While provision for this kind of error is not an elegant justification for ample stack space, there are very clever ways of using the stack for storage of constants (as used in the FORTH language, for example) which sometimes justify the allocation of 'unusual' stack space.

We now initialize our data space. When the computer is turned on, the contents of memory, outside the control of the operating system initialization routines, is unpredictable. If the contents of buffer spaces and other data spaces is critical (control characters could cause program failure) it is best to clear them. We, therefore, clear our data spaces with '0's (SETUP) but our file buffer with 'spaces' (CLBF), which seems more appropriate for a text buffer.

There is one provision not made in this program which I consider essential. You might want to add it yourself. A good programmer will always provide some means of terminating an operation in progress if it may be done so with safety. Operations which do not lend themselves to this auto-termination must certainly include disk I/O, for directory damage can, indeed - will, result. But suppose you began a print cycle and found you had green paper in the printer and wanted violet paper? Unless the system interrupts are disabled, the keyboard will continue to be sampled many times each second, and could intercept your keypress. As a minimum feature, I like to provide each instance of required keyboard input with an 'escape' feature, not necessarily included in the option line. The HOME key could be used for this purpose. As an example, when the Baud Rate option line appears and one wishes to abort the print mode, the HOME key could return the operator to OPTION. The presence and functioning of such a procedure must be mentioned in your instruction manual.

An instructive modification to SOURCE.SRC is the redesign of the file name entry box to allow space for a full file specification. MCHAR will now be set to 15 so the following specification could be entered - 0:XXXXXX.YYY;NN but keep the final cr in case the version number is not entered.

How's your subscription?

Past issues of Colorcue have been delayed in part because subscription renewals came in too late. We are few in number and each renewal is critical to our existance. Keep a close watch on your shipping label and when it reads "0" (or before it reads "0") send your renewal check promptly. Since we plan one full Volume at a time, send $3.00 for each issue of the current Volume due you. (There are six issues per Volume.) We are continuing our guarantee of six full issues or a refund for each unpublished issue. Those of you who were "oversubscribed" at the $2.00/issue rate will have that lower rate honored for VOL VI.

COMPUTER SHOPPER: FASCINATING ADVERTISING CIRCULAR WITH INTERESTING ARTICLES AND NEWS UPDATES. SAVINGS YOU WON'T BELIEVE! ANNUAL SUBSCRIPTION FOR 12 ISSUES IS $15, SENT 3RD CLASS MAIL. EACH ISSUE IS APPROXIMATELY 160 PAGES LONG AND CONTAINS EQUIPMENT, SOFTWARE AND BOOK REVIEWS, APPLICATIONS AND CONSTRUCTION ARTICLES ("HOW TO BUILD YOUR OWN ROBOT"), USER GROUP INFORMATION, AND CLASSIFIED AND COMMERCIAL ADVERTISEMENTS FOR EVERY POSSIBLE COMPUTER-RELATED ITEM. HIGHLY RECOMMENDED! [COLORCUE HAS NO CONNECTION OF ANY KIND WITH COMPUTER SHOPPER.] COMPUTER SHOPPER. PO Box F. TITUSVILLE, FL 32781.

Another modification (which I highly recommend) might be to reformat the display of the 128 bytes so they make labelled fields appropriate to a data base. This was suggested in the last article and some of you have already done so. One reader changed his file extension default to 'ADR' for 'address book.' His field assignments were as follows:

```
NAME FIELD: 40 Characters
STREET ADDR: 40 Characters
CITY FIELD: 28 Characters
STATE & ZIP:  8 Characters
TELEPHONE: 12 Characters

TOTAL = 128
```

It is only a short step from this point in SOURCE.SRC to a working data base program that can search fields for specific ranges of data and display only targeted items.

This completes construction of SOURCE.SRC. Add the modules of Listing V. and test the PRINT and EXIT routines. If you have run into space limitations on your screen editor because you do not have a 32K memory, you may omit the comment fields. The project is hardly over because there is ample room for both the suggested modifications and those you create yourself (the most important ones!) It is true that 90% of your file requirements are contained in this simple program. As with most everything else in this life, success with it will be, primarily, the result of risk.

OTHER DISK ROUTINES. Here is a brief overview of some additional disk routines you can experiment with. There is a provision for 'getting' and 'putting' strings of bytes—or records—in disk files. A record, in this case, is defined as a string of bytes terminated by a line feed or form feed character, and the records are 'stacked' sequentially, one after the other, in the file. You are, of course, not limited to only 128 bytes.

PTREC (16BBH-v9.80,v8.79)(3285H-v6.78) will write such a string to disk. You must follow our previous procedure through CALL OPEN and then set these parameters:
a) Place record buffer address in BC; b) place the record length in DE, and c) place the FPB pointer in HL. d) Call PTREC.

Upon completion, A will be gone, BC will point just pass the last byte written, DE will be 0000H if no errors occurred, and the FPB pointer will still be in HL. If Carry and Zero flags are both set, the end of the file was reached before all bytes were transferred.

Another routine, **PVREC** (16B1H-v9.80,v8.79)(327B-v6.78) will behave in the same way as **PTREC** except that it will write the byte count of the DE registers into the file as the first two bytes, low byte first. These bytes may be retrieved when 'GETting' to simplify local storage or printing.

To 'get' bytes from either record routine above, use **GAREC** (168DH-v9.80,v8.79)(3257H-v6.78). The OPEN and RWSEQI routines must be performed before the first call only, as described in these articles. Then place a) the record buffer address in BC, b) the record buffer length in DE, and c) the FPB pointer in HL. At completion the registers will be as **PTREC** except that the DE register will show how many bytes were read. If both Carry and Parity flags are set a valid terminator was not seen. In this case, the specified number of bytes will be read, but the next call to **GAREC** will start reading at the next byte (and end at the first terminator it sees.)

Beyond the scope of this series are routines for reading and writing 'image' files to and from memory, and routines for reading and writing complete blocks (128 bytes) to disk file. (See Dale Dewey's publication, referenced in the FEB/MAR issue, or the ICS System Listing for details)□

[1] Each character transmission consists of a single 'start' pulse which tells the printer a character is to follow, an eight-bit 'burst' of pulses, representing the ASCII character (with the MSB unused - that is, set to '0') followed by one or two 'high' levels representing the 'stop' bits. These stop bits signal the end of a single character. The detection circuitry in some peripherals is 'fast' enough to be satisfied with one stop bit. Others require a longer pulse. If in doubt, two stop bits should be used. Theoretically, the faster the bits are transmitted, the faster they will print, but once baud rates become higher than 1200, there is usually no significant increase in printing speed with today's printers because the printing mechanism cannot respond that fast. 'BAUD' is an honorary unit of transmission rate, named after Baudot, inventor of an early character set for teletype transmission.

[2] Alert programmers will recognize another way to set baud rate and stop bits. Remembering the sequence 'PLOT 14,27,8,B', where PLOT 14 sets one stop bit and B is a number from 1 to 7, we may place such a sequence in a byte string, LXI H,BAUD and use **OSTR** to send it. If you arrange the byte string as shown here, the '7' (a default value for B) may be replaced by the selected number with an LXI H,BR and MOV M,A. However, conversion tables are important to assembly programming and great fun! If you're new to conversion tables, I suggest staying with the procedure in the listing.

```
BAUDTB: DB 00H,81H,82H,84H,88H
        DB 90H,0AH,0CH
```

If you want two stop bits, change BAUDTB as follows:

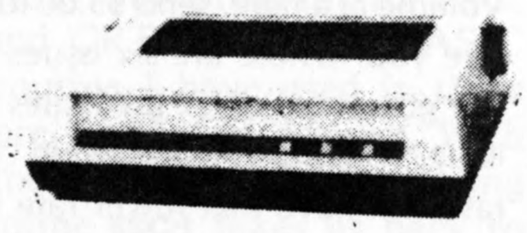
```
BAUD: DB 14,27,18 ;prepare ESC sequence
BR:   DB 7,239    ;add rate and 'end
```

[3] This method of re-entering FCS has engendered some controversy. It will work

=#=	=NAME=	=FUNCTION==========
1	Null	No operation
2-7		Variable
8	BS	Back space print head
9	HT	Horizontal tab/next stop
10	LF	Line Feed
11	VT	Vertical tab/next set line
12	FF	Form Feed/top of form
13	CR	Carriage Return
14-23		Variable
24	CAN	Cancel prior text/this line
25-26		Variable
27	ESC	Escape Command
28-31		Variable

FIG 2. ASCII PRINTER CODES

'Variable' functions provide selection of type face, underline, boldface, elongated characters, graphics and all special functions. This group of codes will vry from printer to printer. Check your own manual for specific instructions.



the first time it's tried. But if you re-enter SOURCE with ESC T and attempt to exit to FCS a second time, it will fail.

[4] The stack provided is excessively generous! As a rule of thumb, for a program of this size, 80 bytes would be more than enough. The FCS stack, by the way, begins at 8044H in the 3651 computer, and at 8042H in the CCII. It works its way back toward screen memory which ends at 7FFF. It is interesting to observe the stack, which may be done with the MLDP (ICS), DBUG (Comtronics), or IDA (Bill Greene). The procedure is to insert a 'break point' at an appropriate place in the program running under the debugging software, and then make a 'dump' of the stack area for examination.. If the stack area is one designated by the user, it may be cleared with '0's before program execution to make tracing easier. □

PRINTER MATERIALS

Here are some sources you may want to tap for printer supplies such as continuous paper, labels, stationery, ribbons, and mailers. Catalogs are available, usually at no charge. If you suscribe to 'Computer Shopper' (see insert in this issue) you will find frequent 'specials' in this same category that can be real money-savers.

QUILL CORPORATION. 100 S. Schelter Road, Lincolnshire, IL 60069. Ribbons, bond papers, printwheels, continuous labels and paper, floppy disks, general stationery supplies.

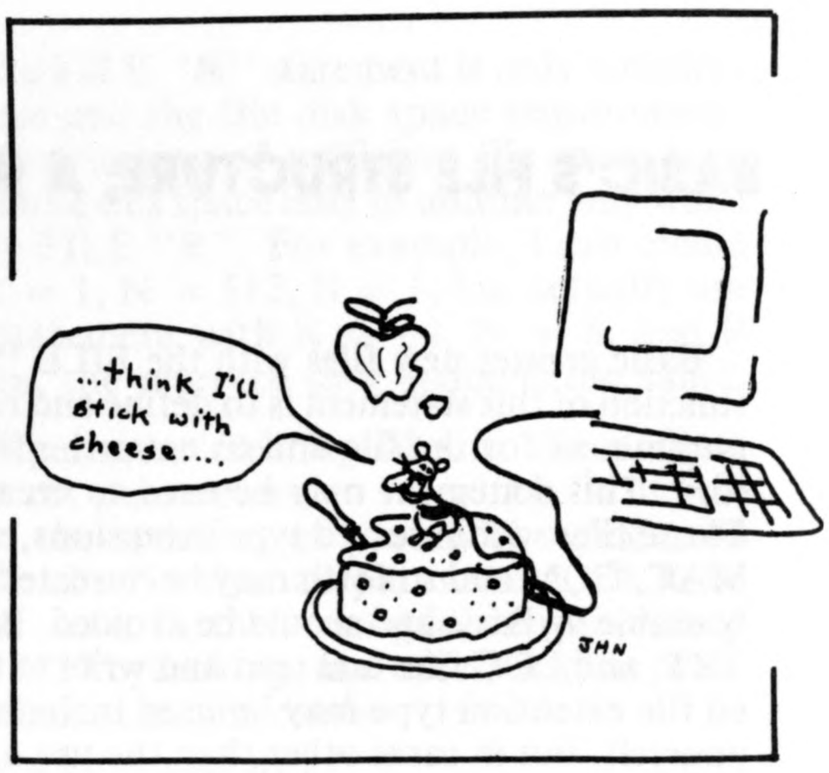
NEBS COMPUTER FORMS. 12 South Street, Townsend, Massachusetts 01469. Continuous printed stationery, envelopes, file cards, labels (you name it!); computer furniture, disks, disk file devices, copystands, ribbons, computer hardware and furniture, business forms.

DELUXE COMPUTER FORMS. 530 North Wheeler Street, PO Box 43046, St. Paul, Minnesota 55164-0046. All kinds of custom forms, including checks, stationery, business utilities, labels, envelopes, mailers. They also sell disks. Somewhat more expensive.

VULCAN BINDER AND COVER. PO Box 29, Vincent, Alabama 35178. This is a great source for looseleaf binders in any variety you might want. They supply plastic sheets for holding disks in a 3-ring binder, disk carrying cases and files, and many other stationery items you're paying a lot more for now.

OBSCO. 11 Dalewood Lane, King Park, New York 11754. Labels and paper at extraordinary savings. Catalog may not be available, but you can telephone them at 516-360-1750. Sample of prices: 5000 labels 3-1/2 X 15/16, one across, continuous form, for $12.95; 1000 sheets continuous 'easy perf' 20– stock printer paper for $17.95.

BCCOMPCO. 800 South 17, Box 246, Summersville, MO 65571. I don't know the extent of their line, but printer ribbon sales are frequently advertised. Sample of recent prices; C. Itoh Prowriter and Epson: (new for C. Itoh) 12 for $96, (new for Epson) 12 for $66, (reloads—you send old cartridges for refill) $6 each for 2 or more, 'off-brand'—12 for $54. They advertise in COMPUTER SHOPPER.



A magazine dedicated to "real world" applications of computers. THE COMPUTER JOURNAL discusses measurement and control, interfacing, hardware construction, robotics and EPROMS, according to the literature. $24/year for 12 issues. Send inquiries to THE COMPUTER JOURNAL: PO Box 1697C, Kallspell, MT 59903.

MORROW USERS GROUP: A GROUP EXPLORING THE POTENTIAL OF THE MORROW MICRO DECISION HAS FORMED IN ORANGE, CONNECTICUT. CMDUG (CONNECTICUT MICRO DECISION USERS GROUP) MEETS ONCE EACH MONTH. A QUARTERLY, "CMDUG NEWSLETTER" COMES WITH THE MEMBERSHIP FEE OF $12 PER YEAR. FOR MEMBERSHIP AND DETAILS OF MEETINGS, WRITE DAVE MINTIE. CMDUG, 226 BOSTON POST ROAD , ORANGE, CT 06477.

COLORCUE BOOK SERVICE: Several major bookstores carry a line of computer books. Among these are Walton, B. Dalton, Doubleday, and Computerland stores. If you are having trouble gettings books and have exhausted your local resources, Colorcue will try to get them for you. We charge the price of the book plus a handling and mailing fee that runs between $2.00 and $5.00. You will be billed after the book(s) are sent. Just write to us with the author, title, publisher and number of copies.

NETWORK SUBSCRIBERS! Here's a beginning!]

COLORCUE: CompuServe 71106, 1302

Frepost Computers: Source TC1251, Micronet* 70210, 374

Intelligent Computer Systems: Source TCB610

David Suits, Ben Barlow: CompuServe 70045, 1062

Christopher Zerr: CompuServe 71445, 1240

*MicroNET is a part of CompuServe dedicated to personal computer user functions and activities. There is a surcharge for use of this service.

BASIC'S FILE STRUCTURE; A Review

Basic creates new files with the FILE "N" statement. The function of this statement is to define and reserve the disk space parameters for the file and to enter the file in the disk directory. This statement may be used to create any type of file. Those files with reserved type extensions, such as LDA, PRG, MAC, COM and so forth may be "created" but are not readily usable by Basic and should be avoided. Basic can create SRC, TXT, and DOC files and read and write to them. Any unreserved file extension type may be used including ones you specify yourself, but in cases other than the use of RND, the file will be a sequential file, best containing only ASCII characters. Non-ASCII characters read by Basic can cause disruption of Basic and should therefore be avoided. As a "reserved" file type, RND files will be recognized and treated in a special way by Basic.

The file to be created must not already exist on the disk in the addressed drive. By "exist" we mean that at least one parameter among file name, type and version number must be unique. If the file name and type already exist on the disk, Basic will create another like file with a higher version number.

The FILE "N" Statement has this general form -

FILE "N",F$,R,N,B

where F$ is the file name—a group of quoted ASCII characters, R is the number of records in the file, N is the number of bytes in each record, and B is the number of records that will be read into computer memory at one time.

The file name may have from one to six characters drawn from the alphabet "A" to "Z" and the numerals "0" through "9". The file name may not contain spaces or any other ASCII character not in the list above. Valid characters may be arranged in any order; for example, these are all valid file names —

12GOT.RND 4.RND IF56K.TRP I1TU.NOW 0.HAB

The file name may end with a specific version number if you wish to specify it. You may also preceed the file name with a disk drive specification, contained in the F$ string. For example —

F$ = "0:SAMPLE.FRK;09" or
F$ = "CD1:FROTH.CLD;1A"

Note that version numbers are in hexadecimal. The maximum number of versions varies with the system software. You may determine your maximum number through trial and error.

The number of records, R, to be contained in the file is a decimal number between 1 and 32767. For random files (RND), a "record" implies that you will construct a sequence of bytes which have meaning as a group, and that this same meaning is reflected in each and every record in your file. For example, a record might consist of a name, address and telephone number, meaning that every record will have these same data groups (and in the same sequence and each of the same field length). If you wish to store no more than 100 sets of names-addresses-telephones then you will assign the value 100 to R. For non-random files, a record may be anything you wish, with the requirement that each record will have the same defined maximum length. whether this entire length is used or not.

Creating the File: The FILE "N" Statement.

The number of bytes to be held in each record, N, is a decimal number between 1 and 32767. This number will be made large enough to contain the longest record you expect to store. If the "name" field can be no longer than 30 characters, the "address" field 40 characters, the "telephone number" field 12 characters, then the value assigned to N will be at least 82—all these fields in the same record.

When choosing the value of N, consider that Basic creates file space in integer multiples of the standard disk block of 128 bytes. If you assign a value of 1 to R, and 125 bytes to N, one entire disk block of 128 bytes will be assigned to your file, the last 3 bytes being "wasted." With the same single record, and a value of 129 bytes for N, two entire disk blocks will be assigned to your file with 127 bytes "wasted." A "wasted" byte cannot be used by another file.

When the value of R is greater than 1 however, bytes within a single disk block or several disk blocks may be shared among the records. Suppose R = 3 records and N = 85 bytes per record. The total number of bytes for three records is 3 * 85 = 255. This file will fit into two disk blocks (2 * 128 = 256) with one "wasted" byte. You can see how many "wasted" bytes exist in a disk block by examining the LBC (last block count) hexadeciaml number in the file directory. LBC tells you how many bytes are used in the last block of a file. If the number is 80H (=128) then every byte has been used. In this example, nothing would be gained by making N=84, because then three more bytes would be "wasted" in the disk blocks assigned to this file.

The blocking factor for the file, B, is a decimal number between 1 and 255 that infers how many blocks of file data you wish to be read into computer memory at one time, and, therefore, how much computer memory you wish dedicated to holding your file data. The maximum number 255 means that no more than 255 * 128, or 30640, data bytes may be read into computer memory at one time. Just as Basic allocates file space in integer multiples of the 128-byte disk block, so does it read blocks in multiples of 128 bytes. You cannot read only 67 bytes of data from disk into computer memory, nor can you read only 129 bytes of data into computer memory. Any bytes unused (not subject to GET) will still be in computer memory if they lie within a partially-specified 128-byte block.

When assigning the blocking factor it is customary to choose a number that will coincide with an integer number of records, and, at the same time, an integer number of disk blocks. If N=128 bytes, then each integer value for B will bring that number of records into computer memory. If N=64, then a value of 2 for B will bring two records into computer memory at one time. If N=64 you cannot have a blocking factor of 1 (=64 bytes—an error message will result) since disk blocks can only be read in integer multiples of 128. For N=64, B must be in multiples of 2 (2 * 64 = 128 bytes).

As the value for B determines the amount of user computer memory you want dedicated to file data storage (B * N) it also defines an additional memory space which will hold the file access parameters required by Basic. The Com-

pucolor Manual cites a formula for computing the exact memory space requirement. If there is sufficient room in user memory for many files, then the only penalty for reading a large number of records into computer memory is the disk access time. If all the records will not be needed (as they might be for procedures such as sorting) the needed record may be accessed without bringing all previous records into memory. (See FILE "R" next issue.) In summary, the smallest disk access time will be achieved by assigning a value to B that brings only the required number of records into computer memory.

All parameters in FILE statements may be expressed as variables. You may assign numbers to R, N, and B and a string to F$ and state FILE "N",F$,R,N,B to open a file.The string file name, F$, may be a concatenation of strings, and R,N, and B may be members of numeric arrays as in these examples:

FILE "N",N$ + T$,L(3), K(2),MW(20), or
FILE "N",STR$(D) + T$,J(X),YM(A,B),BY(R,T,U)

Remember that the FILE "N" statement is only concerned with the file name and the file disk space requirement. If N * B permits the reserving of sufficient file space then you are free to distribute this space later in another way when the file is opened by FILE "R". For example, I can create TEST.RND with R = 1, N = 512, B = 1, but actually use it in a FILE "R" statement with R = 64, N = 8, and B = 16. In both cases, the reserved disk space is the same.

It is helpful to experiment creating files in "immediate mode", using different parameters, to see which Basic will accept. Apart from the specifics of parameters discussed here, you must only observe that all parameters, F$,R,N, and L are specified. As a further example, if you wished to create a SRC file 5120 bytes long, and access it all at once, then the create statement might read—

FILE "N","TEST.SRC",1,5120,1.

Next time we will continue with a discussion of the FILE "R" statement. □

NEXT ISSUE:Using Morrow MicroDecision with the CCII; An inexpensive pen plotter you can use; ASCII, Masks and BCD; Review of Basic's file structure; Simple encryption in assembly language; Peter Hiner's FASBAS and more!

Back issues of COLORCUE contain a wealth of practical information for the beginner as well as the more advanced programmer, and an historical perspective on the CCII computer. Issues are available from October 1978 to current.

DISCOUNT: For orders of 10 or more items, subtract 25 % from total after postage has been added. **POSTAGE:** for U.S., Canada and Mexico First Class postage is included; Europe and South America add $1.00 per item for Air Mail, or $ 0.40 per item for surface; Asia, Africa, and the Middle East add $ 1.40 per item for Air Mail, or $ 0.60 per item for surface. **SEND ORDER** to Ben Barlow, 161 Brookside Drive, Rochester. NY 14618 for VOL I through VOL V; and to Colorcue, 19 West Second Street, Moorestown, NJ 08057 for VOL VI and beyond.

1978	**VOL I**	**$3.50 each**
	No. 1-3:	OCT/ NOV/ DEC
1979	**VOL II**	**$3.50 each**
	No. 1-3:	APR/MAY/JUN
	No. 4-5:	JAN/FEB/MAR
	No. 6-7:	AUG/SEP/OCT
	No. 8:	NOV XEROX COPY, $2.00
1980	**VOL III**	**$1.50 each**
	No. 1	DEC/JAN
	No. 2:	FEB
	No. 3:	MAR
	No. 4:	APR
	No. 5:	MAY
	No. 6:	JUN/JUL
1981	**VOL IV**	**$2.50 each**
	No. 0:	DEC/JAN
	No. 1:	AUG/SEP
	No. 2:	OCT/NOV
1982	No. 3:	DEC/JAN
	No. 4:	FEB/MAR
	No. 5:	APR/MAY
	No. 6:	JUN/JUL
	VOL V	
	No. 1:	AUG/SEP
	No. 2:	OCT/NOV
1983	No. 3:	DEC/JAN
	No. 4:	FEB/MAR
	No. 5:	APR/MAY
	No. 6:	JUN/JUL
1984	**VOL VI**	**$3.50 each**
	No. 1:	JAN/FEB

UNCLASSIFIED ADVERTISEMENTS

FOR SALE: Compucolor II, v6.78, 32K, extended keyboard, manuals including Programming, Maintenance, "Color Graphics" and "Basic Training." 15 disks included. Excellant condition. Asking $1000. Art Tack. 1127 Kaiser Road, SW, Olympia, WA 98502.

FOR SALE: Compucolor II computer, v8.79. Basic keyboard only, 32K memory, switchable lower case character set, CRT filter, handshake option installed. Very good condition. $800. Joseph Norris c/o The David Hafler Company, 5910 Crescent Blvd, Pennsauken, NJ 08109. 609-662-6355. (No, I'm not abandoning ship—I still have three more!)

The following items are from the estate of Myron T. Steffy.

FOR SALE: Two Compucolor II computer systems, v6.78. Each computer has full keyboard, 32K memory, dual disk drives, Devlin Analog Protector, 2 Devlin Ram Cards with software switch, lower case character set, and character generator for FREDI. One of the above units has the Comtronics updated ROM. Excellant condition. $1200 each.

Morrow MicroDecision CP/M computer with dual disk drives, 64K memory and software package. No monitor. $900.

Novation CAT, 300 Baud modem, Votrax unit, TI SR52 calculator and TI programmer's calculator, 2 MFT transfer switches for RS232 outputs. All in good condition. Make offer.

Diablo Model 630 letter quality printer. Cost $1850 wher new. Excellant condition. Make offer.

Address inquiries to Bill Shanks. 1345 West Escarpa, Mesa AZ 85201. 602-962-0130.